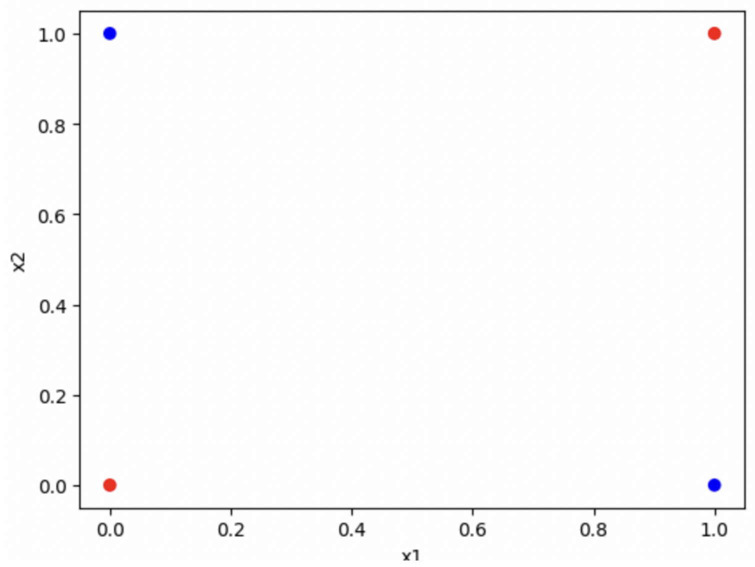
Zeynep Kurtulus
240037253
Assignment1_Part3_Report

Q1. Why is it important to use a random set of initial weights rather than initializing all weights as zero in a Neural Network? [2 marks]

If we initialize all the weights as zero in a neural network it would result in all the neurons seeing the same input point and doing the same calculations which would result in the neurons learning the exact same thing. This is a problem for us because all neurons will end up having identical behaviors which would mean that they will not learn any unique features from the data and all the neurons in a layer will learn the same thing. Accordingly, neurons will miss to catch comptex patterns in the data which will make the system much less efficient. When we assign random weights to each neuron we are preventing them to have the same behavior, we are making them unique from the start so that our neural network is able to handle different complex problems.

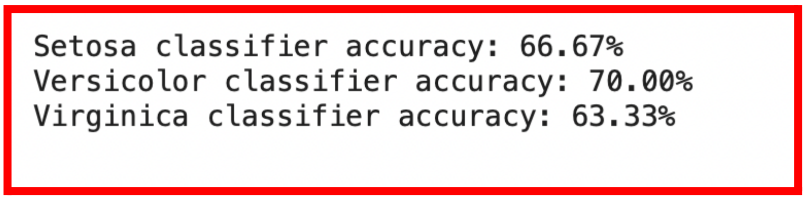
Q2. How does a NN solve the XOR problem? [1 marks]

The XOR problem cannot be solved with the linear classification algorithms that we have seen so far because it is a non-linear classification problem. As can be seen from the graph below we cannot draw a linear decision boundary to separate the two classes, that's why xor is considered to be a non-linear classification problem.

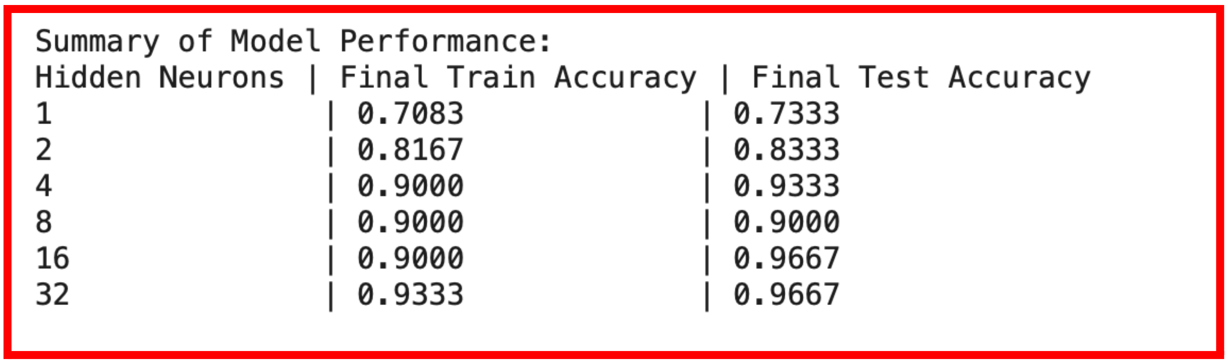


By using neural networks, we can combine decision boundaries at each level of the network, when we combine multiple layers the network can combine these simpler decision boundaries into a more complex boundary that better separates the data. Each of the neurons will produce something linear but when we combine them we will end up having something more complex than linear which will help us solve the XOR problem. We can have a neuron that implements the AND behavior and a neuron that implements the OR behavior, then when we combine them in the Neural Network we will have implemented the XOR behavior.

Q3. Explain the performance of the different networks on the training and test sets. How does it compare to the logistic regression example? Make sure that the data you are referring to is clearly presented and appropriately labeled in the report. [8 marks]



```
Setosa classifier accuracy: 66.67%
Versicolor classifier accuracy: 70.00%
Virginica classifier accuracy: 63.33%
```



```
Summary of Model Performance:
Hidden Neurons | Final Train Accuracy | Final Test Accuracy
1               | 0.7083               | 0.7333
2               | 0.8167               | 0.8333
4               | 0.9000               | 0.9333
8               | 0.9000               | 0.9000
16              | 0.9000               | 0.9667
32              | 0.9333               | 0.9667
```

As shown in the second table above, networks with 1 and 2 hidden neurons have lower accuracy on both the training and test sets, indicating underfitting. Models with only 1 or 2 neurons are too simple to capture the complexity of the data, resulting in limited ability to learn meaningful patterns. Consequently, the accuracy remains low for both training and test sets.

When we increase the number of hidden neurons to 4, 8, and 16, we observe a significant rise in both training and test accuracy. The additional hidden neurons allow the model to capture more complex patterns in the data, resulting in higher generalization ability and improved test accuracy. These networks achieve a balance between model complexity and generalization, avoiding both underfitting and overfitting.

With 32 hidden neurons, training accuracy shows a slight increase, while test accuracy remains high at 96.67%. This could indicate overfitting, where the model becomes highly adapted to the training data but cannot perform well on unseen data. Overfitting occurs when the model learns the training data too welland that it struggles to generalize to new, unseen samples. This typically results in very high training accuracy but lower performance on the test set.

When examining the logistic regression model's accuracy (see the first table), it performs significantly worse than the neural network models with more than 2 hidden neurons. Logistic regression, due to its linear nature, is limited in its ability to make accurate classifications in datasets that require non-linear decision boundaries, as is the case with the Iris dataset. The neural networks, on the other hand, can capture non-linear patterns, and as a result, achieve test accuracies of up to 96.67%.

In summary, neural networks with 4–16 hidden neurons perform very well, achieving high accuracy on both training and test sets, indicating that these configurations provide an optimal balance for this classification task. Using 32 hidden neurons slightly increases the risk of overfitting, as indicated by high training accuracy but relatively stable test accuracy.

Overall, logistic regression falls short in capturing the complexity of the Iris dataset, as evidenced by its lower individual class accuracies in the range of 63–70%, compared to the neural networks, which achieve significantly better performance by learning non-linear patterns.

Zeynep Kurtulus
Student id: 240037253